



Instituto Politécnico Nacional

Escuela Superior de Cómputo



Desarrollo de Sistemas Distribuidos.

Tarea 5. Multiplicación de matrices utilizando objetos distribuidos.

Nombre: Flores Castro Luis Antonio.

Profesor: Pineda Guerrero Carlos.

Grupo: 4CV13.

2 mayo 2023

Desarrollo.

Creación de grupo de recursos en Azure.

Para la creación del grupo de recursos, dentro del menú principal vamos al menú “Grupos de recursos”.

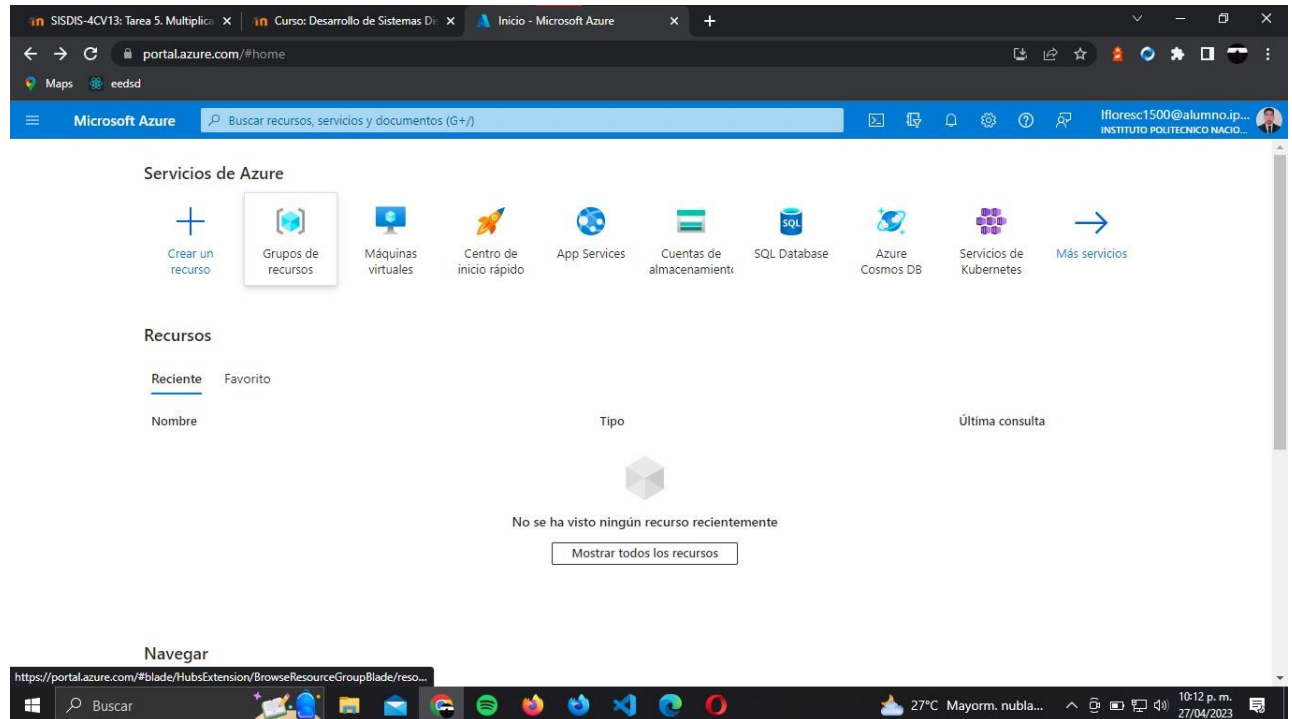


Figura 1. Menú “Grupo de recursos”.

Dentro de este menú, ahora damos click en el botón “Crear grupo de recursos”.

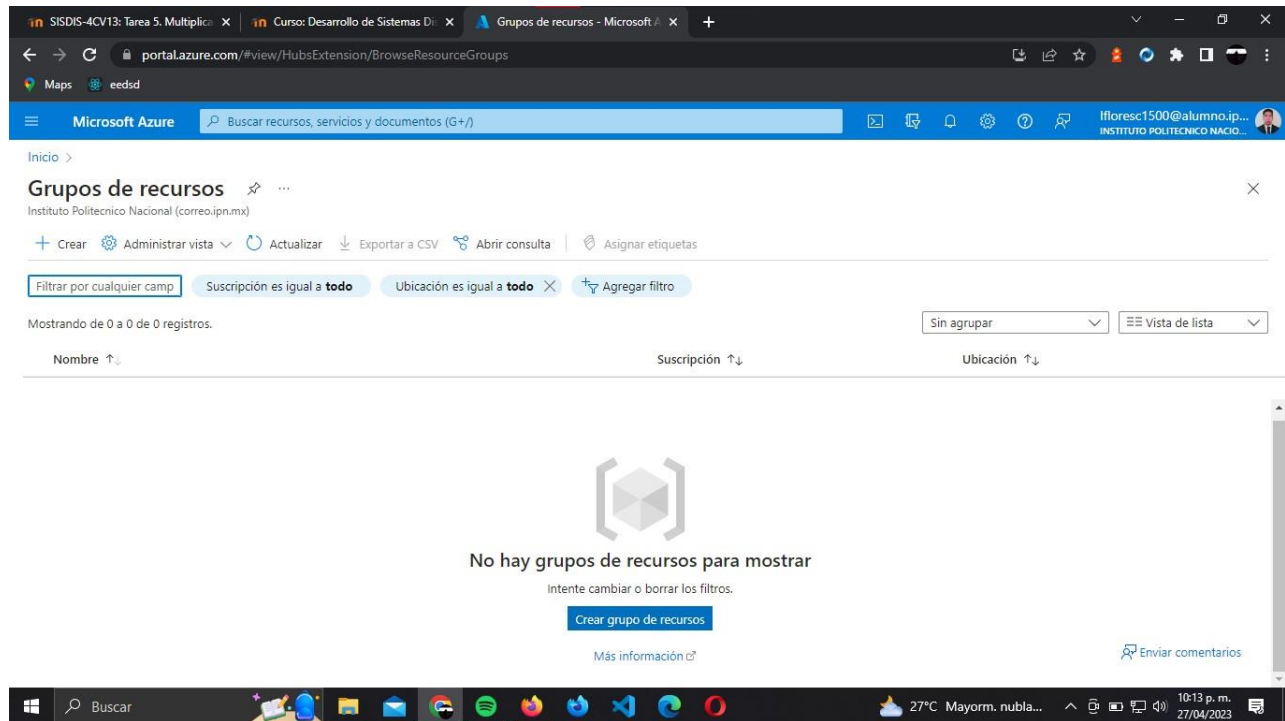


Figura 1.1 Crear grupo de recursos”.

Aquí, se asignará un nombre al grupo de recursos en este caso “Tarea-5” y se elegirá la región en la que se alojará este grupo, después damos click en el botón “Revisar y crear”.

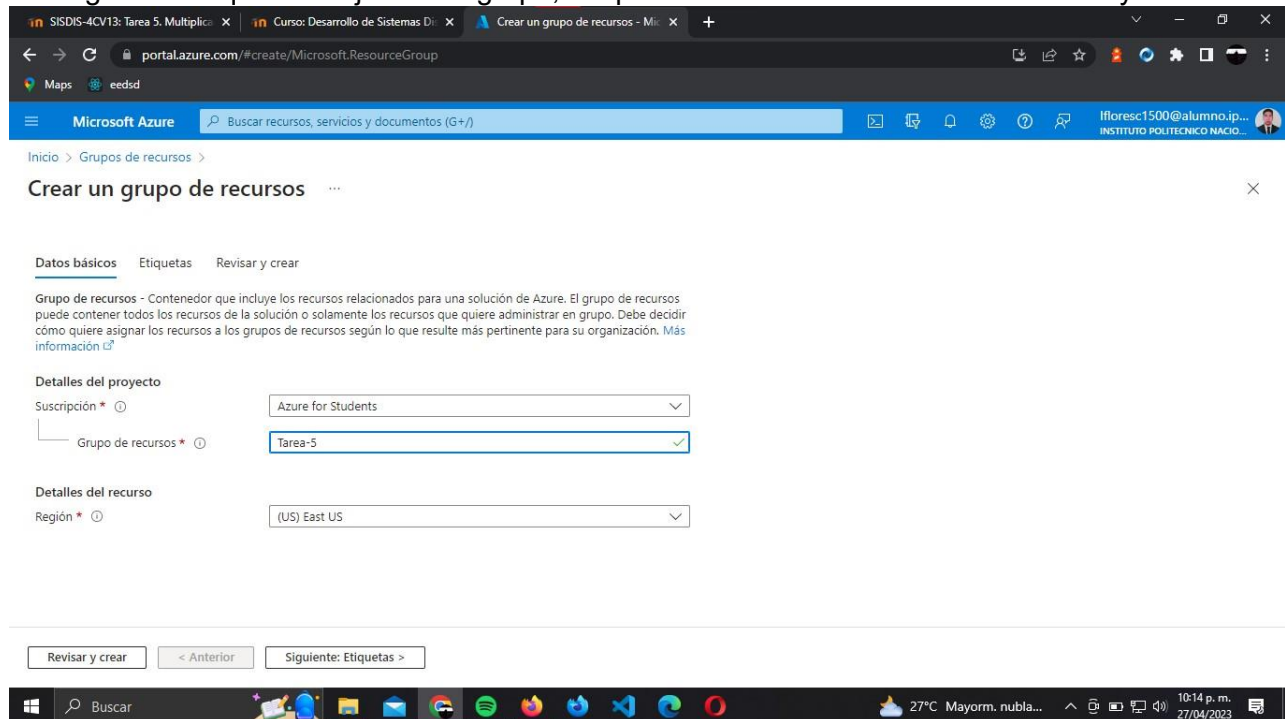


Figura 1.2 Configuración de grupo de recursos”.

Se mostrarán los detalles del grupo de recursos, si son correctos damos click en el botón “Crear”.

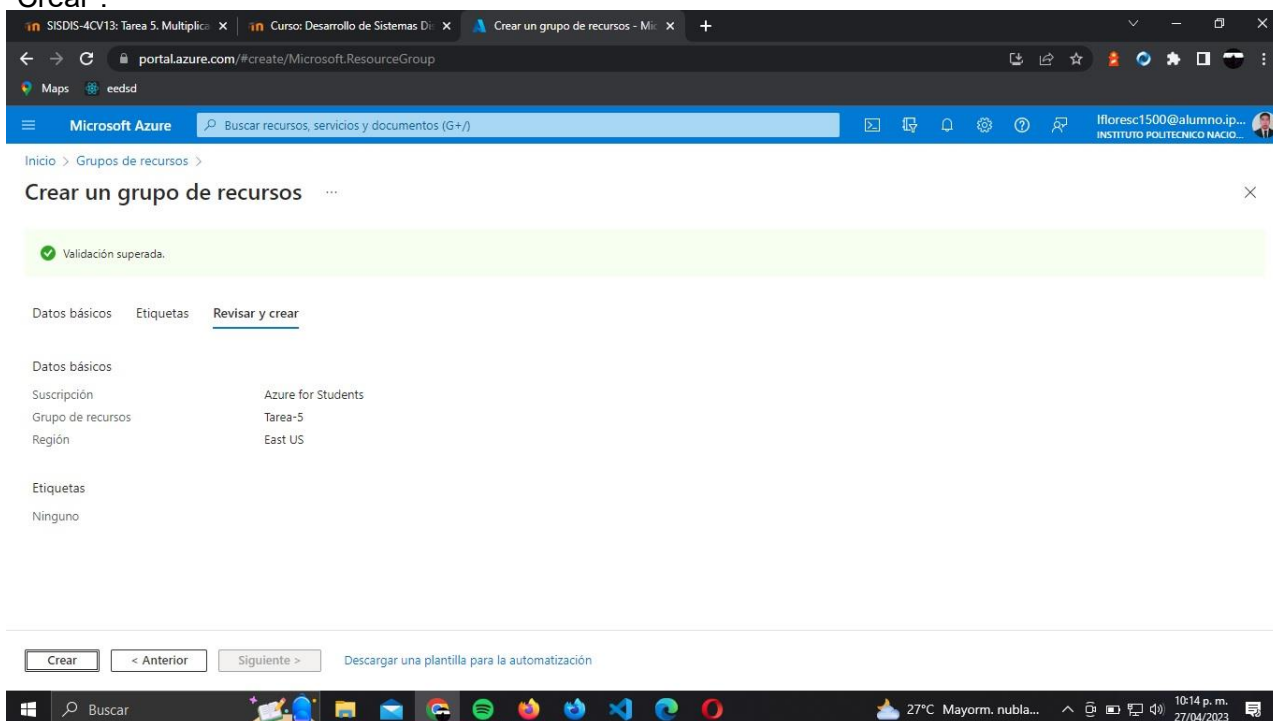


Figura 1.3. Configuración de grupo de recursos”.

Finalmente vemos que el grupo ha sido creado.

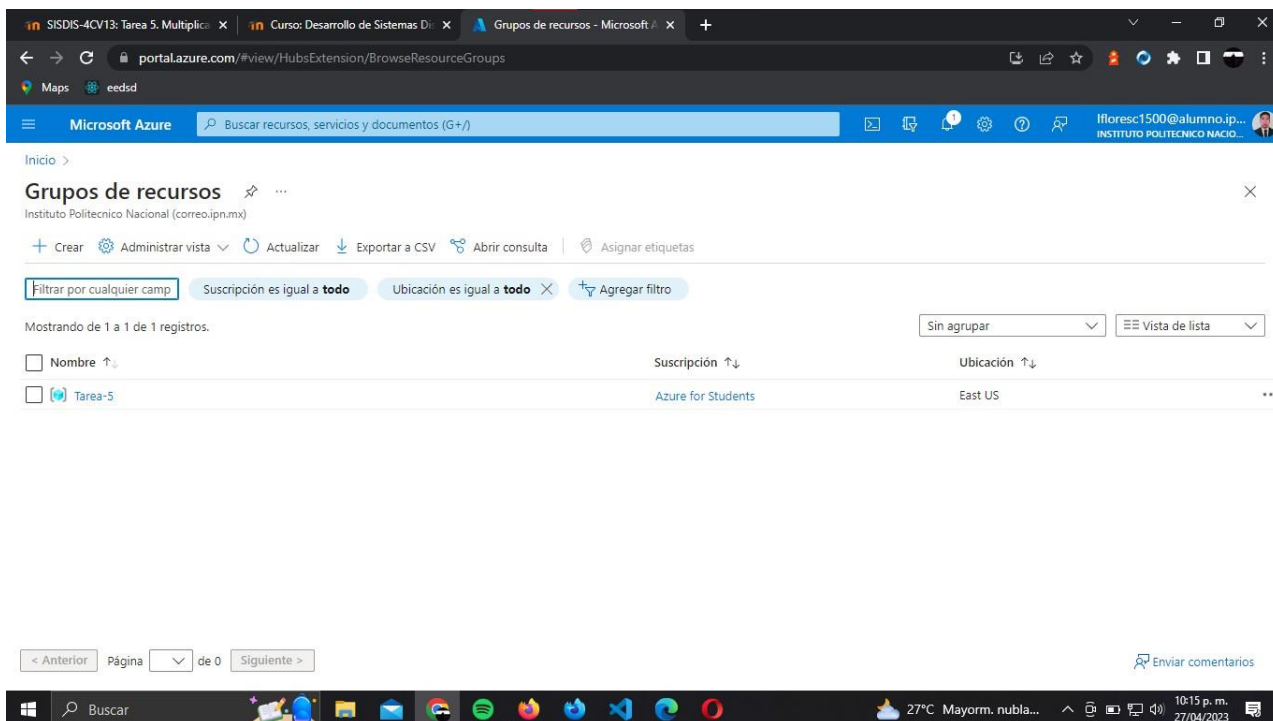


Figura 1.4. Creación del grupo de recursos.

Para la creación de las máquinas virtuales, en el menú principal vamos al menú “Máquinas virtuales”.

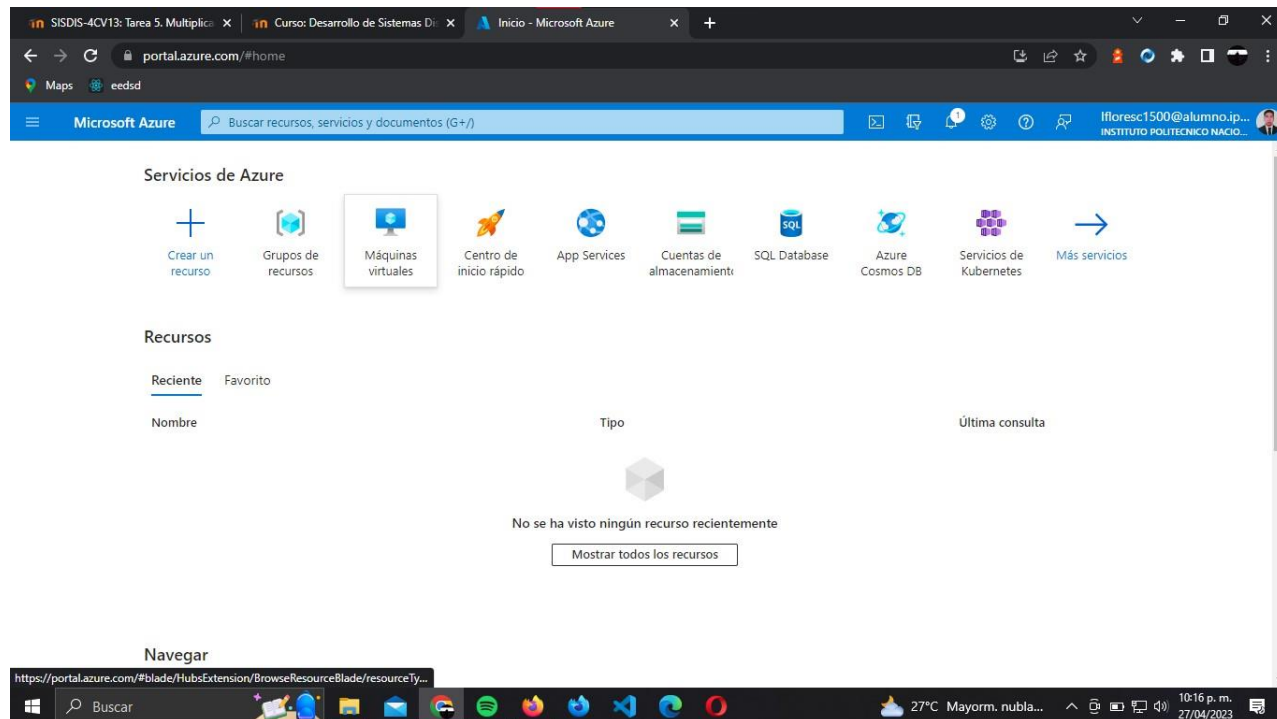


Figura 1.5. Menú máquinas virtuales.

Elegimos el botón “Crear”.

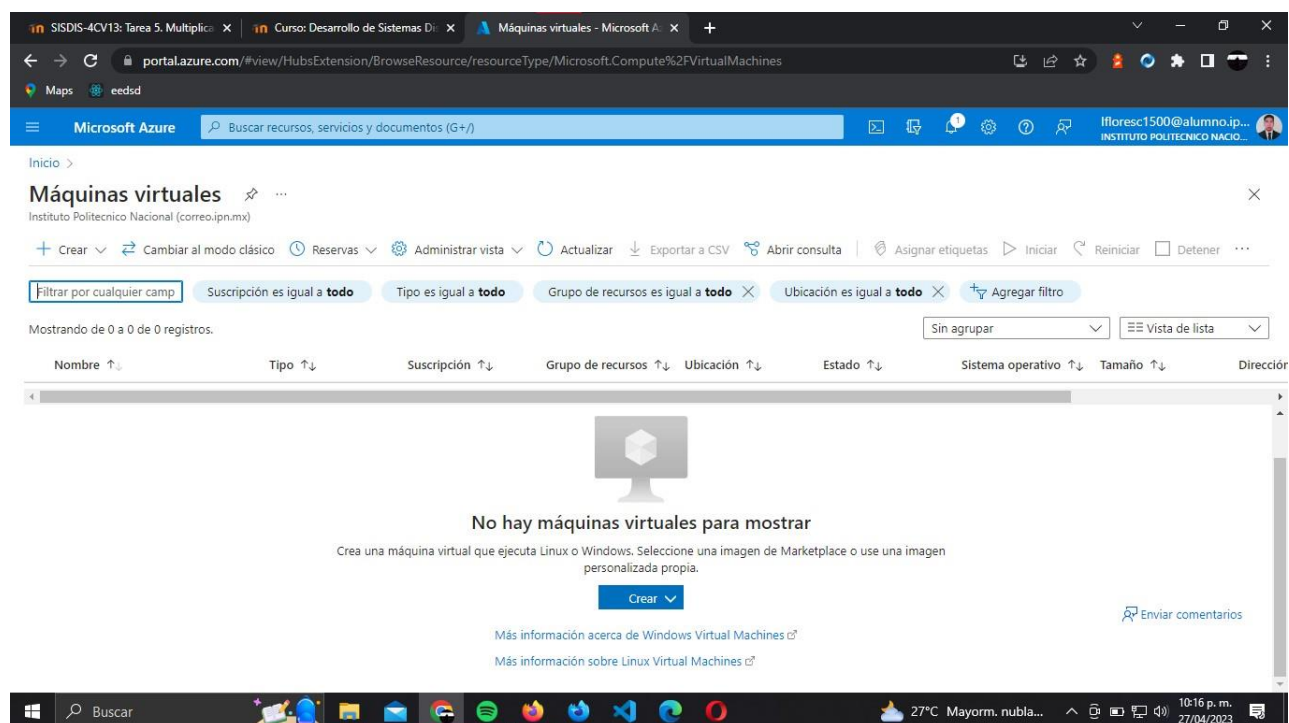
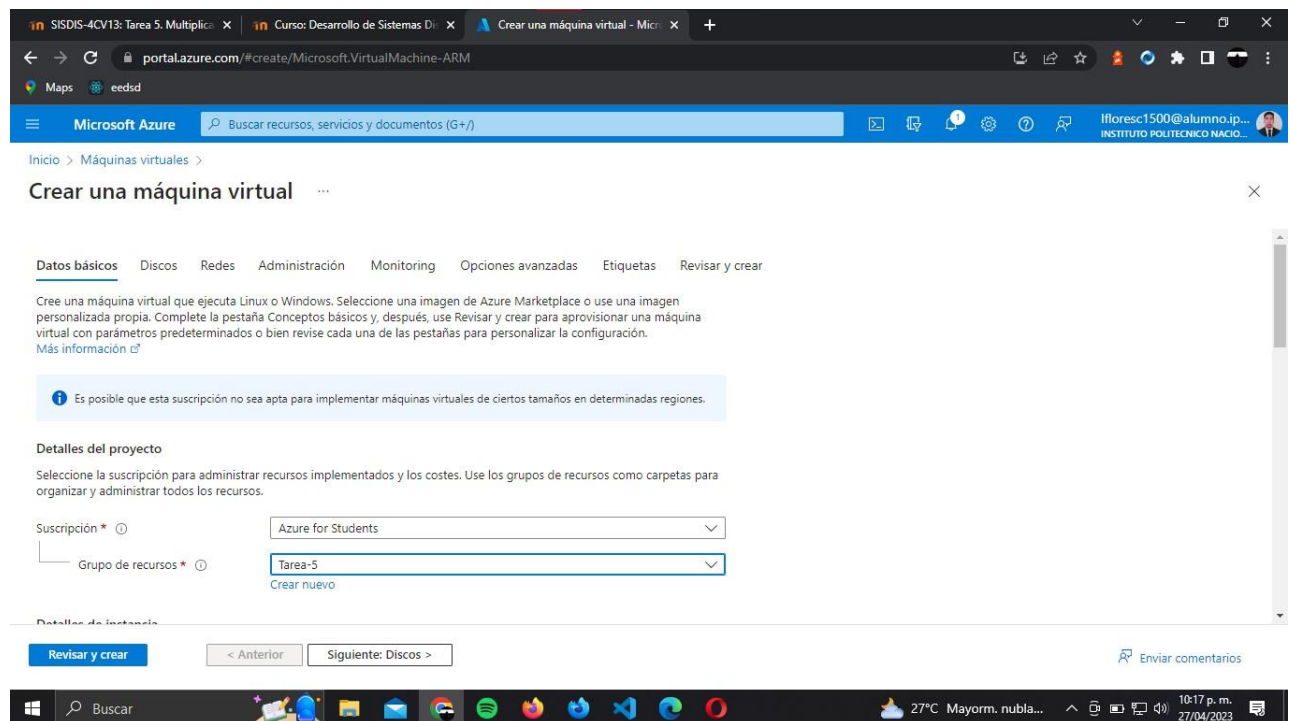


Figura 1.6. Crear máquina virtual.

Dentro del menú, “Datos básicos” escogemos el grupo de recursos que ya se ha realizado.



Microsoft Azure

Inicio > Máquinas virtuales >

Crear una máquina virtual

Datos básicos | Discos | Redes | Administración | Monitoring | Opciones avanzadas | Etiquetas | Revisar y crear

Cree una máquina virtual que ejecuta Linux o Windows. Seleccione una imagen de Azure Marketplace o use una imagen personalizada propia. Complete la pestaña Conceptos básicos y, después, use Revisar y crear para aprovisionar una máquina virtual con parámetros predeterminados o bien revise cada una de las pestañas para personalizar la configuración. [Más información](#)

Es posible que esta suscripción no sea apta para implementar máquinas virtuales de ciertos tamaños en determinadas regiones.

Detalles del proyecto

Seleccione la suscripción para administrar recursos implementados y los costes. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción *

Grupo de recursos *
[Crear nuevo](#)

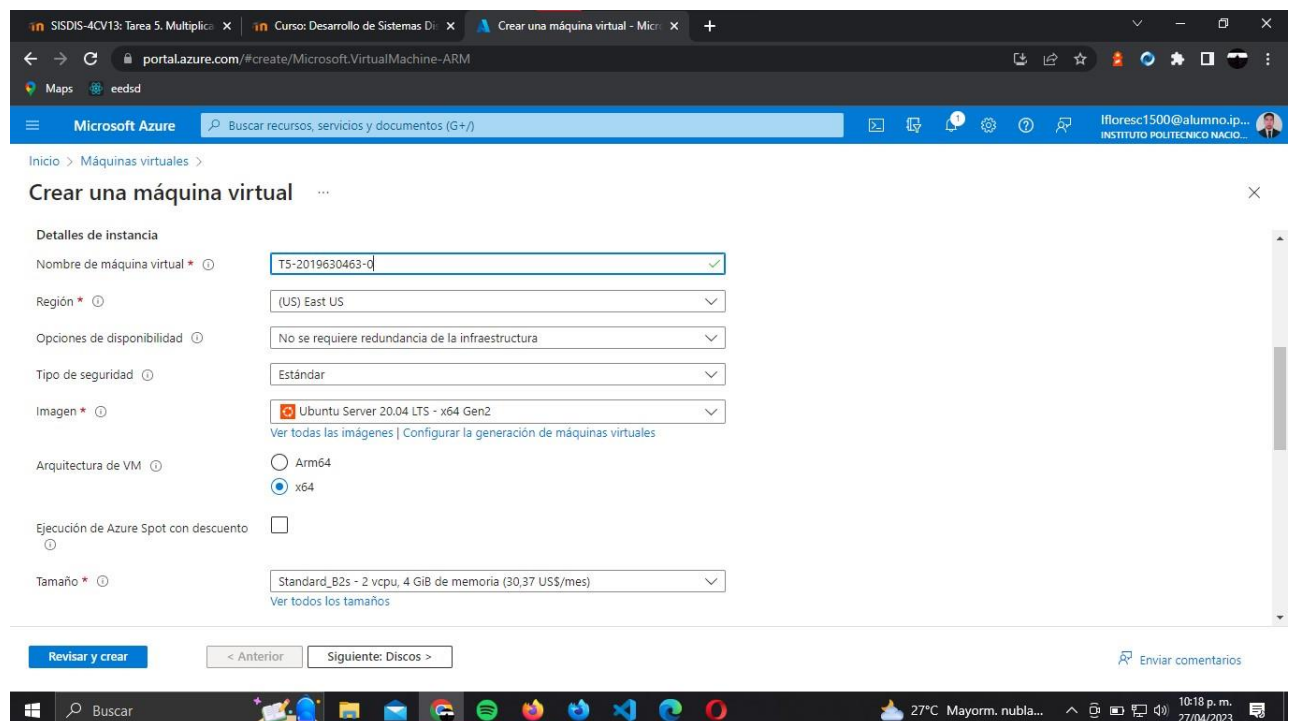
Detalles de instancia

[Revisar y crear](#) < Anterior | Siguiente: Discos >

Enviar comentarios

Figura 1.7. Datos Básicos de máquina virtual.

Ahora para la máquina virtual 0, se le asigna el nombre “T5-2019630463-0”, se elige Ubuntu Server 20.04 LTS, como imagen, arquitectura de x64 y 4GiB de memoria.



Microsoft Azure

Inicio > Máquinas virtuales >

Crear una máquina virtual

Detalles de instancia

Nombre de máquina virtual *

Región *

Opciones de disponibilidad

Tipo de seguridad

Imagen *
[Ver todas las imágenes](#) | [Configurar la generación de máquinas virtuales](#)

Arquitectura de VM ☐ ☒

Ejecución de Azure Spot con descuento ☐

Tamaño *
[Ver todos los tamaños](#)

[Revisar y crear](#) < Anterior | Siguiente: Discos >

Enviar comentarios

Figura 1.8. Datos Básicos de máquina virtual.

Elegimos Contraseña como “Tipo de autenticación” y asignamos nombre de usuario “Nodo0” y una contraseña, de igual forma dejamos el puerto SSH(22) como puerto de entrada, ahora vamos al menú “Discos”.

Figura 1.9. Datos Básicos de máquina virtual.

Aquí, escogemos el tipo de disco HDD estándar para ahorrar saldo.

Figura 1.10. Discos de máquina virtual.

Después, vamos a redes, aquí dejamos la configuración por default.

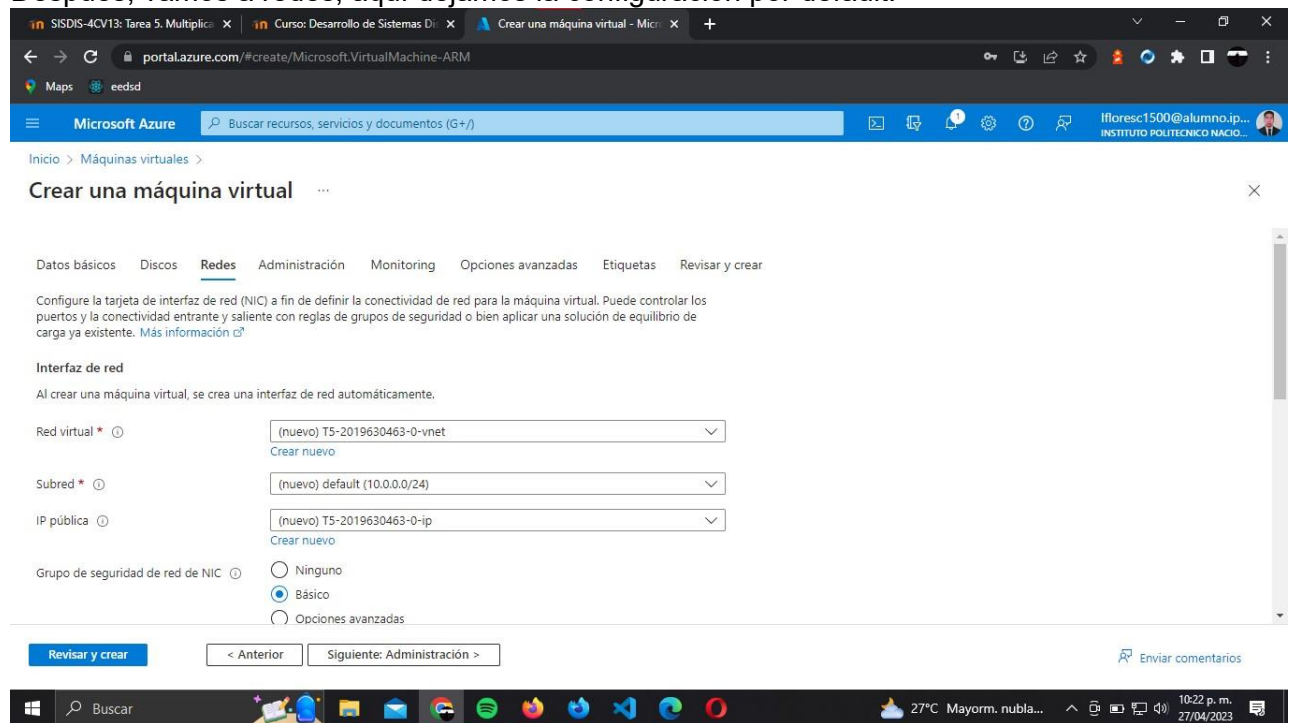


Figura 1.11. Redes de máquina virtual.

En el menú Administración y Monitoring dejamos la configuración que contienen ambos menús.

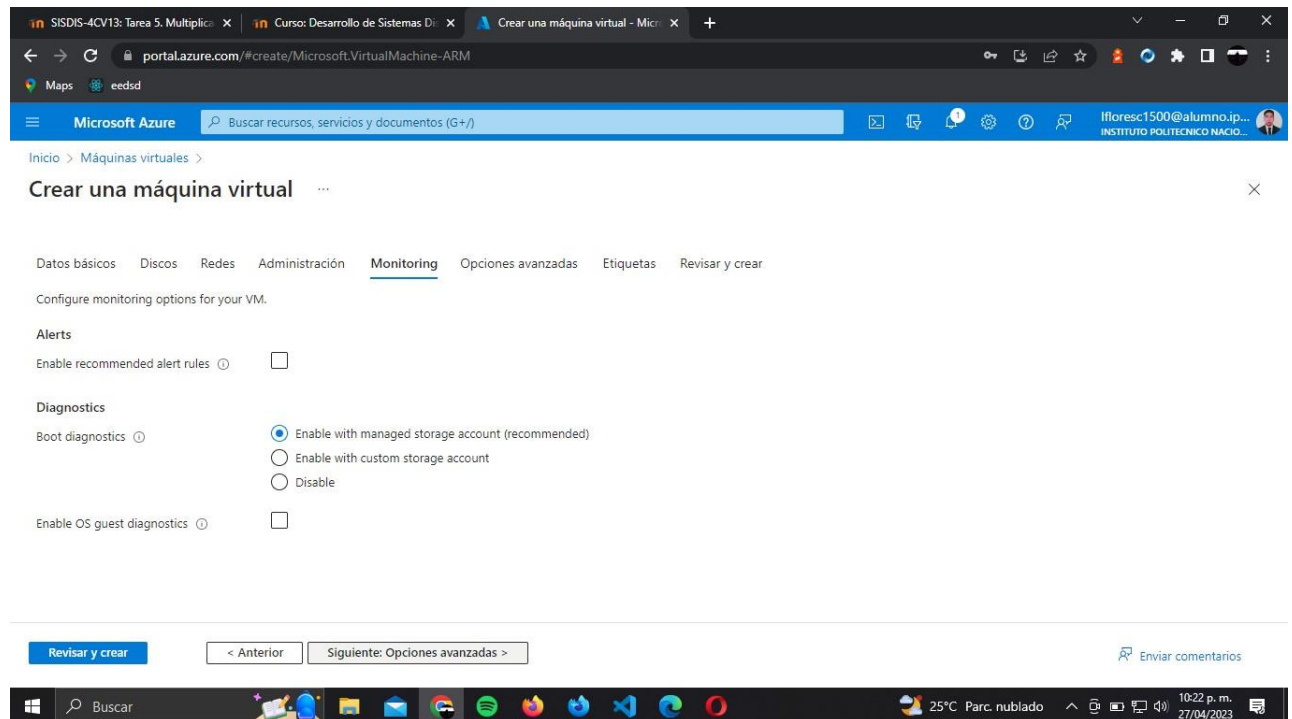


Figura 1.12. Administración y Monitoring de máquina virtual.

En el menú “Opciones avanzadas”, conservamos la configuración inicial.

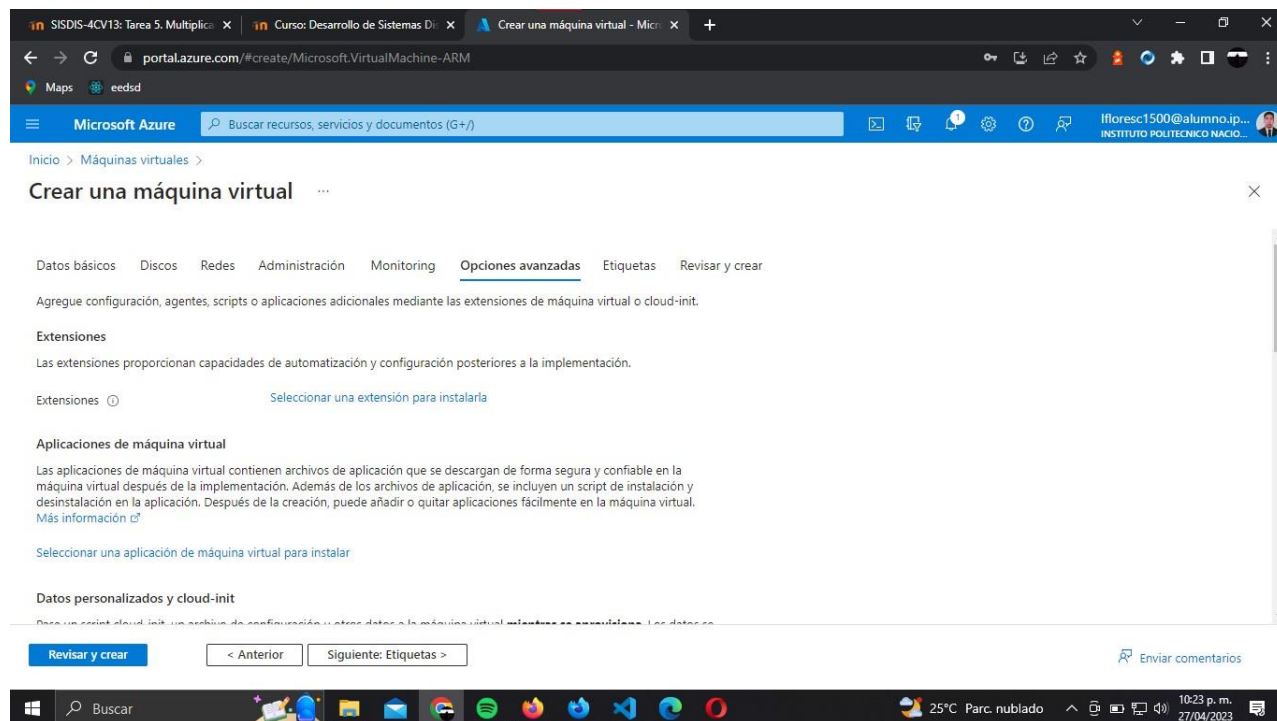


Figura 1.13. Opciones avanzadas de máquina virtual.

De la misma forma, del menú etiquetas no modificamos absolutamente nada.

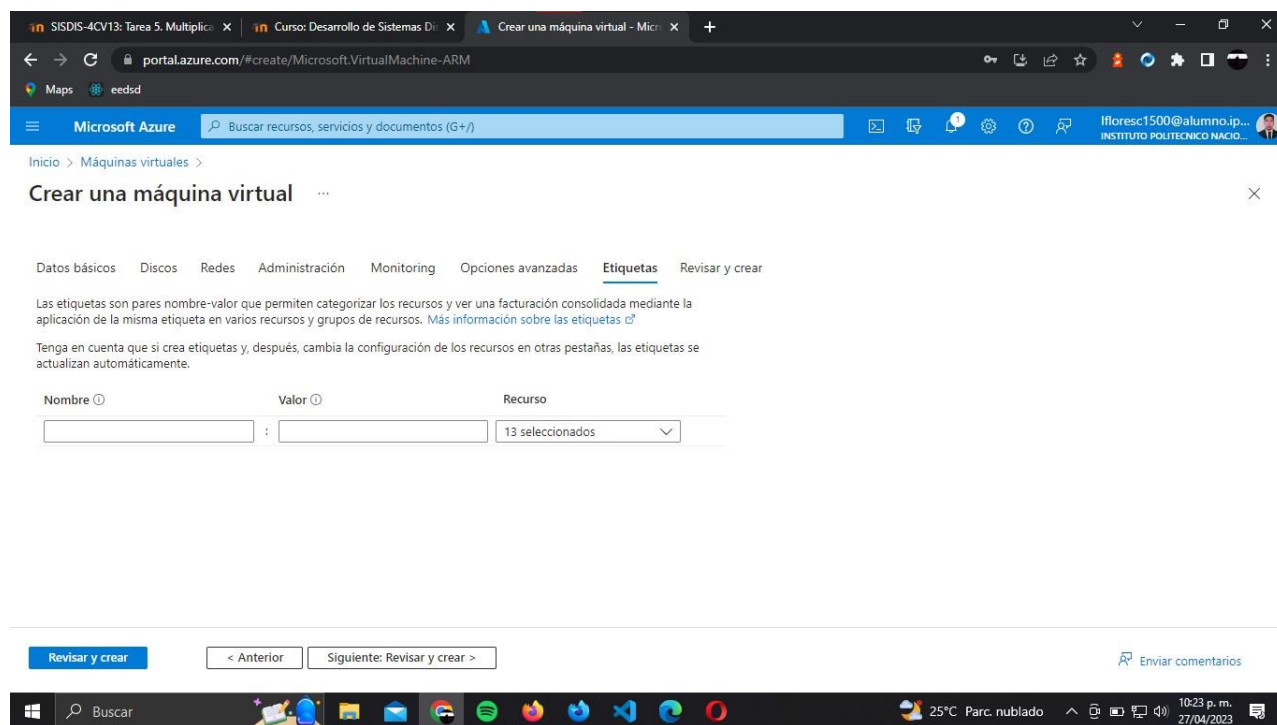


Figura 1.14. Etiquetas de máquina virtual.

Finalmente, en el menú “Revisar y crear” vemos los detalles de configuración y damos click en el botón “Crear” si todos los datos son correctos.

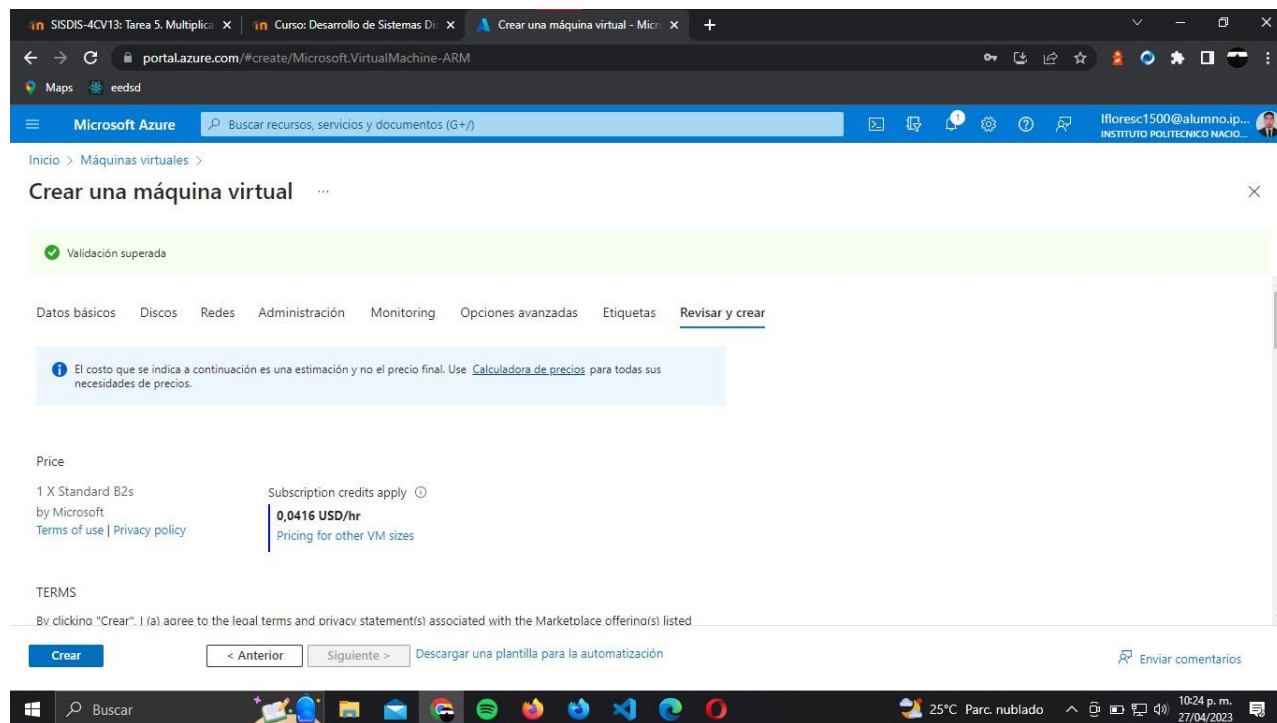


Figura 1.15. Creación de máquina virtual.

Se desplegará una pantalla indicando que esta en progreso, la creación de la máquina virtual.

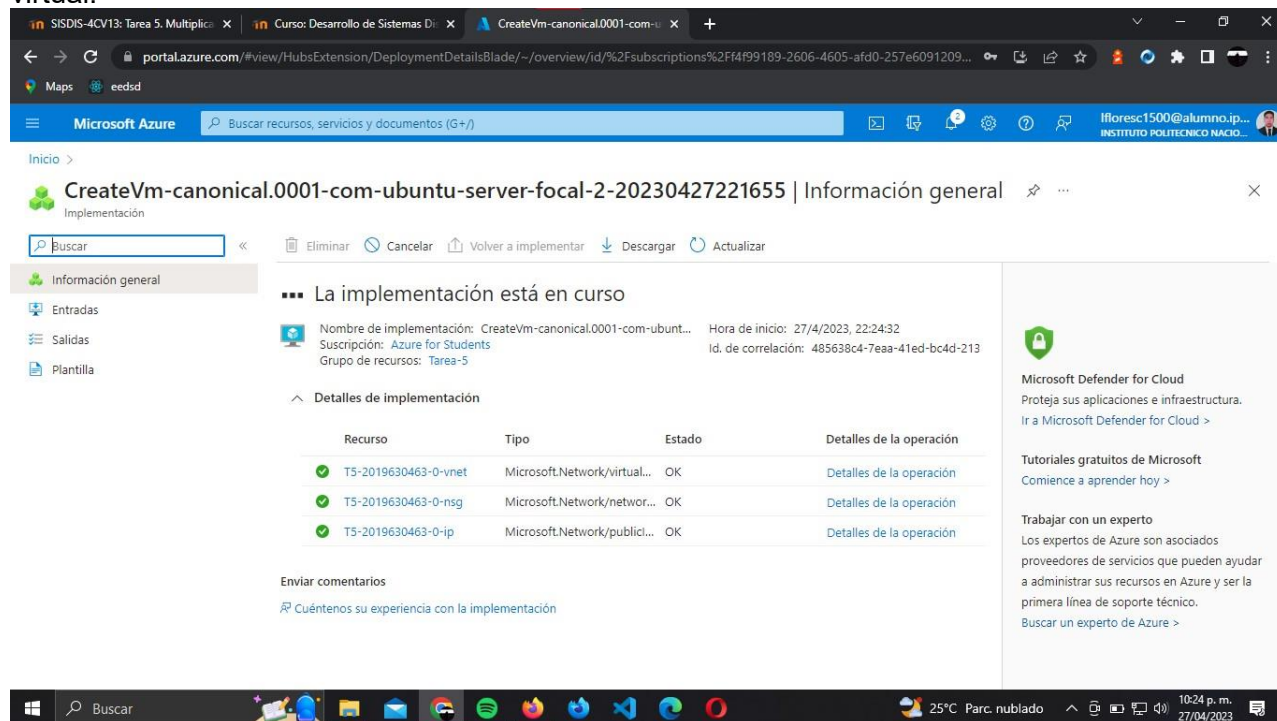


Figura 1.16. Creación de máquina virtual.

Después, aparecerá un botón “Ir al recurso”, damos click en el y se mostrará todos los detalles de la máquina virtual.

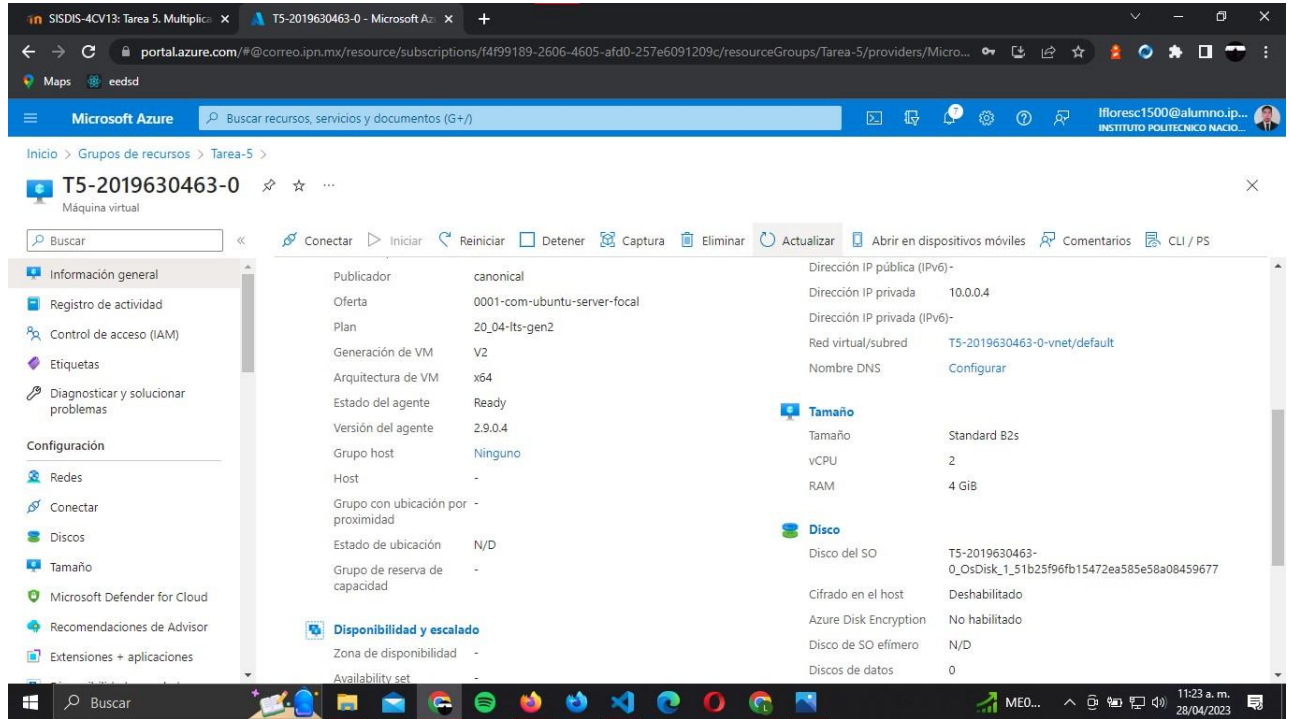


Figura 1.17. Detalles de máquina virtual.

Una vez creadas las máquinas virtuales, ahora accedemos a cada una de ellas mediante el comando `ssh Usuario@IP`, en la IP colocamos la IP publica de cada nodo.

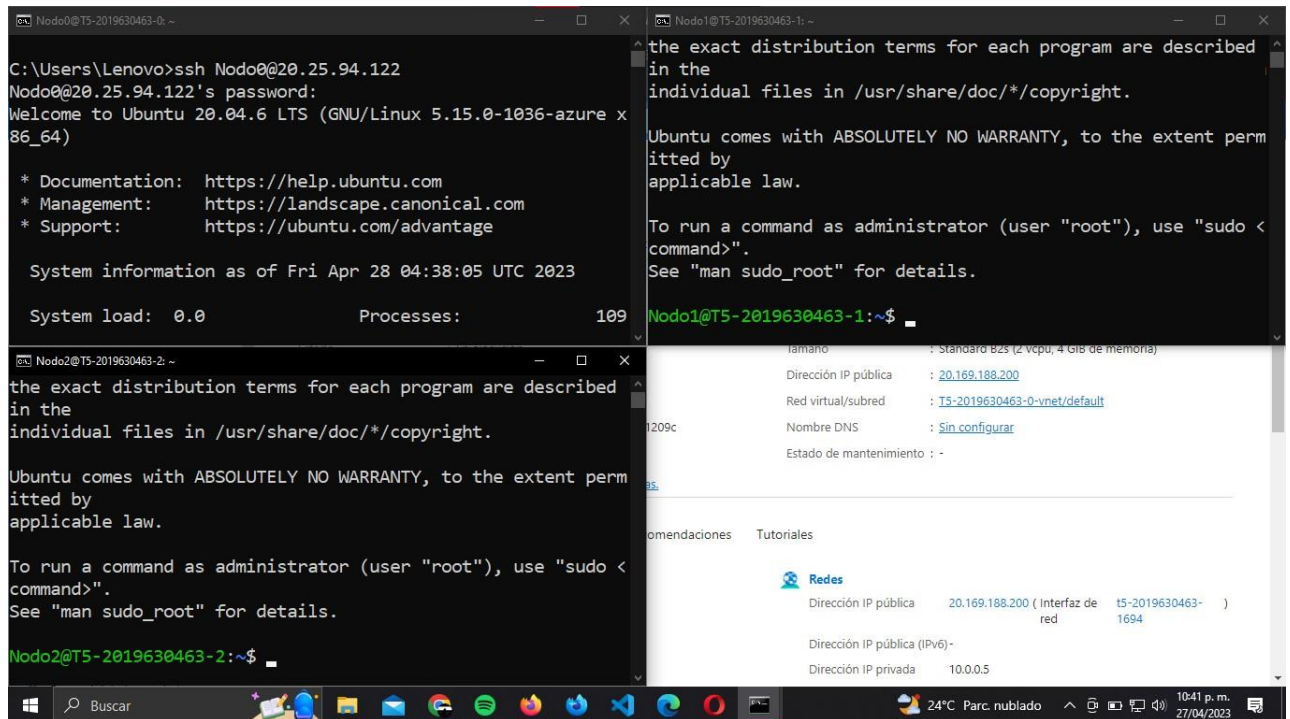
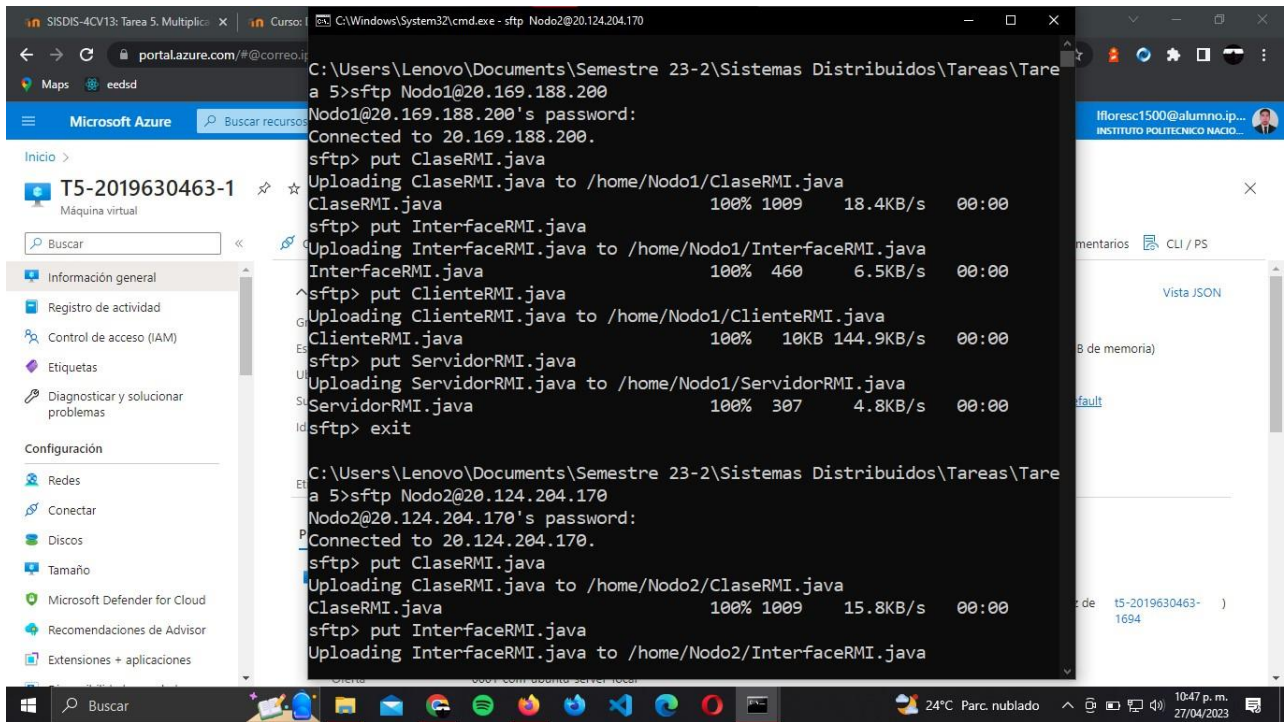


Figura 2. Acceso a máquinas virtuales mediante ssh.

Ahora, mediante el comando `sftp Usuario@IP`, realizamos la transferencia de archivos a cada uno de los nodos, en el caso de los nodos 1 y 2 se envían los programas `ServidorRMI.java`, `InterfaceRMI.java` y `ClaseRMI.java` para el nodo 0, todos los anteriores programas más el `ClienteRMI.java`.



```
C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tare
a 5>sftp Nodo1@20.169.188.200
Nodo1@20.169.188.200's password:
Connected to 20.169.188.200.
sftp> put ClaseRMI.java
Uploading ClaseRMI.java to /home/Nodo1/ClaseRMI.java
ClaseRMI.java          100% 1009   18.4KB/s   00:00
sftp> put InterfaceRMI.java
Uploading InterfaceRMI.java to /home/Nodo1/InterfaceRMI.java
InterfaceRMI.java      100% 460    6.5KB/s   00:00
sftp> put ClienteRMI.java
Uploading ClienteRMI.java to /home/Nodo1/ClienteRMI.java
ClienteRMI.java        100% 10KB 144.9KB/s  00:00
sftp> put ServidorRMI.java
Uploading ServidorRMI.java to /home/Nodo1/ServidorRMI.java
ServidorRMI.java       100% 307    4.8KB/s   00:00
sftp> exit

C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tare
a 5>sftp Nodo2@20.124.204.170
Nodo2@20.124.204.170's password:
Connected to 20.124.204.170.
sftp> put ClaseRMI.java
Uploading ClaseRMI.java to /home/Nodo2/ClaseRMI.java
ClaseRMI.java          100% 1009   15.8KB/s   00:00
sftp> put InterfaceRMI.java
Uploading InterfaceRMI.java to /home/Nodo2/InterfaceRMI.java
```

Figura 2.1. Transferencia de archivos a máquinas virtuales mediante sftp.

Vemos que los archivos se han transferido con éxito en los nodos.

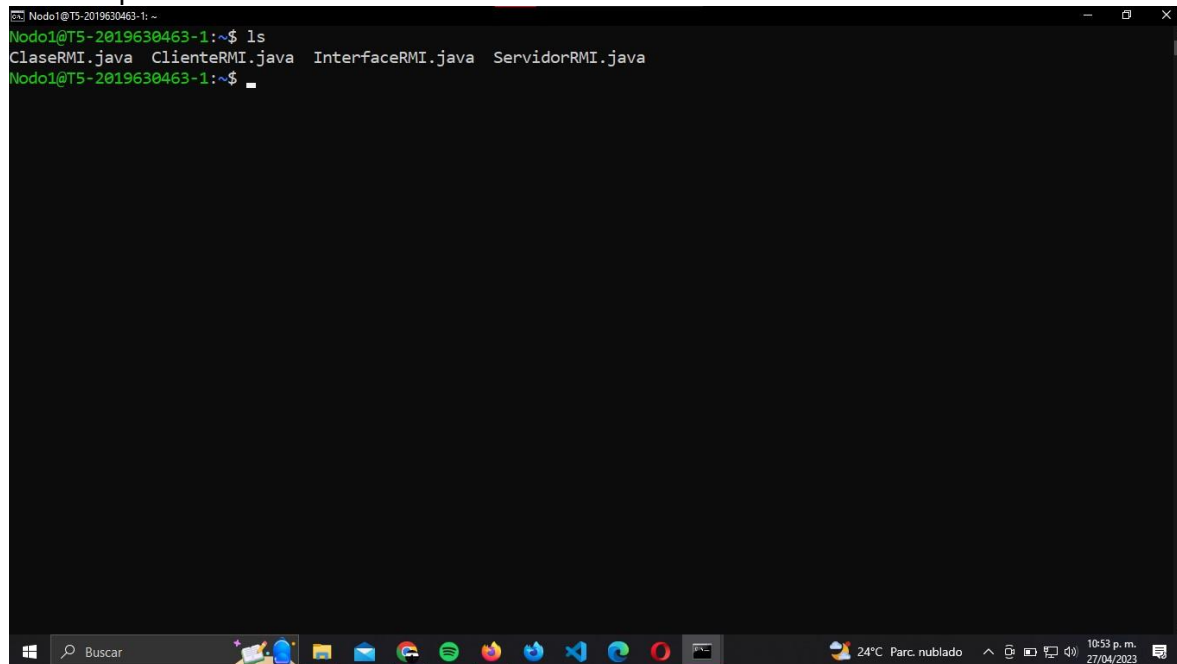
A terminal window titled 'Nodo1@T5-2019630463-1: ~' showing the output of a 'ls' command. The output lists four files: 'ClaseRMI.java', 'ClienteRMI.java', 'InterfaceRMI.java', and 'ServidorRMI.java'. The terminal window is part of a desktop environment with a taskbar at the bottom showing various application icons and system status information like '24°C', 'Parc. nublado', and the date '27/04/2023'.

Figura 2.2. Instalación de JDK en máquinas virtuales.

Se instala jdk en cada uno de los nodos.

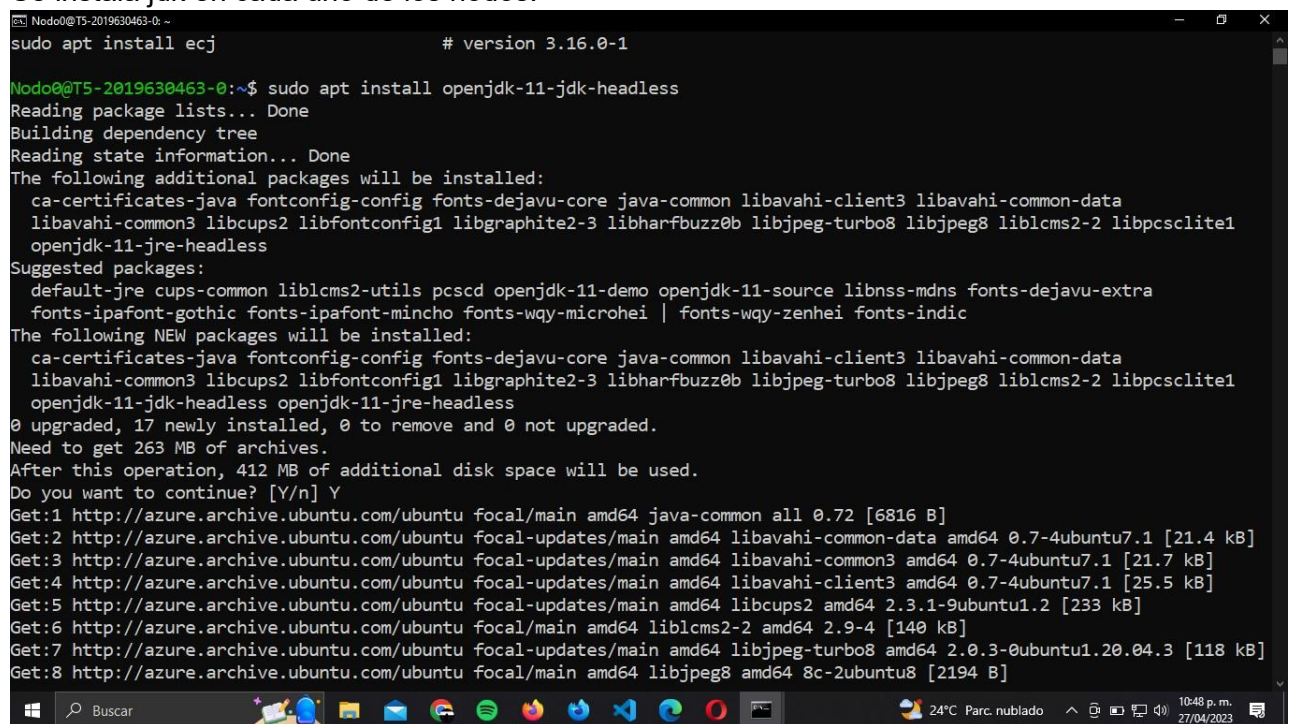
A terminal window titled 'Nodo0@T5-2019630463-0: ~' showing the installation of 'openjdk-11-jdk-headless'. The output includes the command 'sudo apt install ecj', the version '3.16.0-1', and the installation progress. It lists additional packages to be installed, suggested packages, and the disk space requirements. The installation is confirmed with 'Y' for 'Do you want to continue?'. The terminal window is part of a desktop environment with a taskbar at the bottom showing various application icons and system status information like '24°C', 'Parc. nublado', and the date '27/04/2023'.

Figura 2.3. Instalación de JDK en máquinas virtuales.

Ahora se compilan todos los archivos transferidos a cada uno de los nodos.

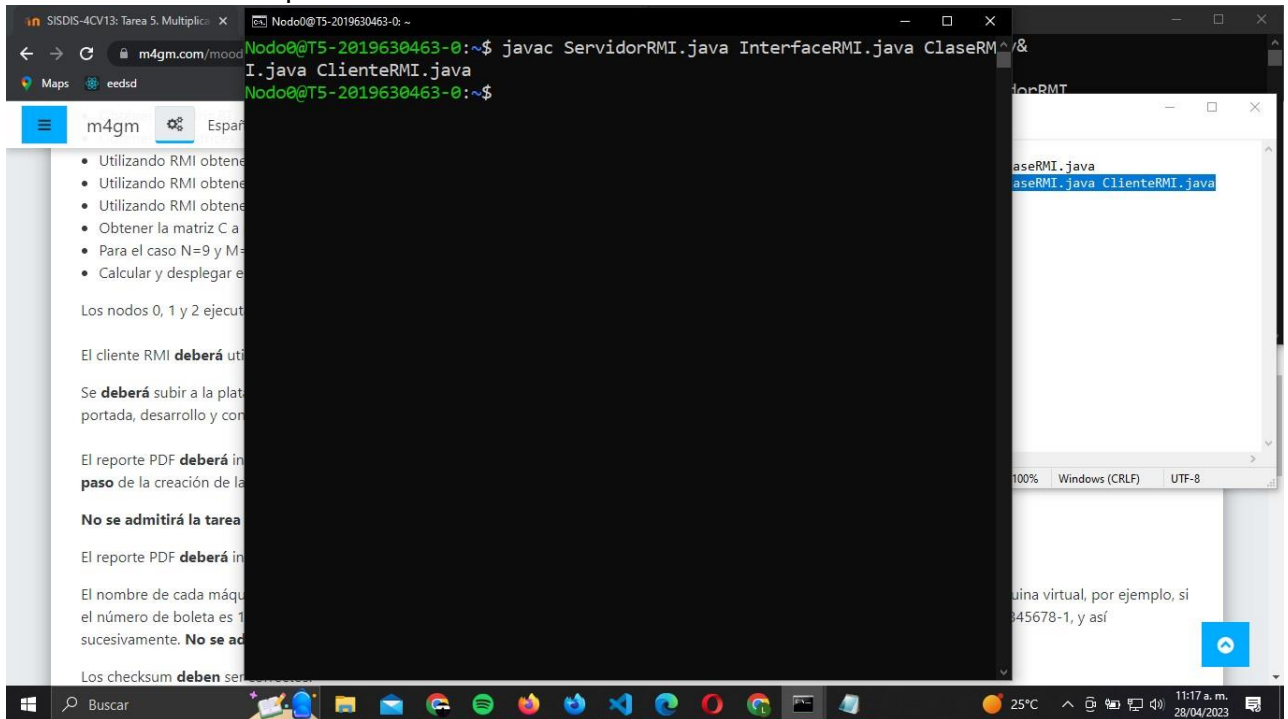


Figura 2.4. Compilación de los programas en las máquinas virtuales.

Ya compilados los programas, ahora se ejecuta "rmiregistry&" y "ServidorRMI.java" en cada uno de los nodos.

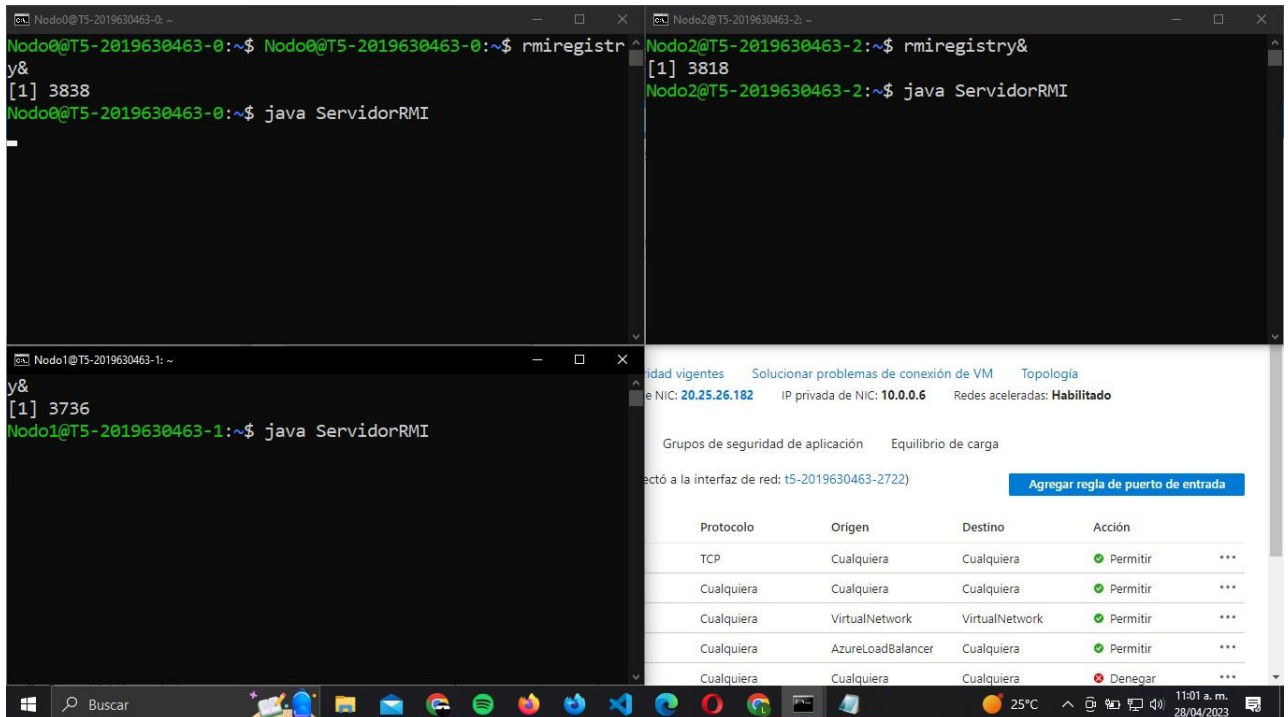
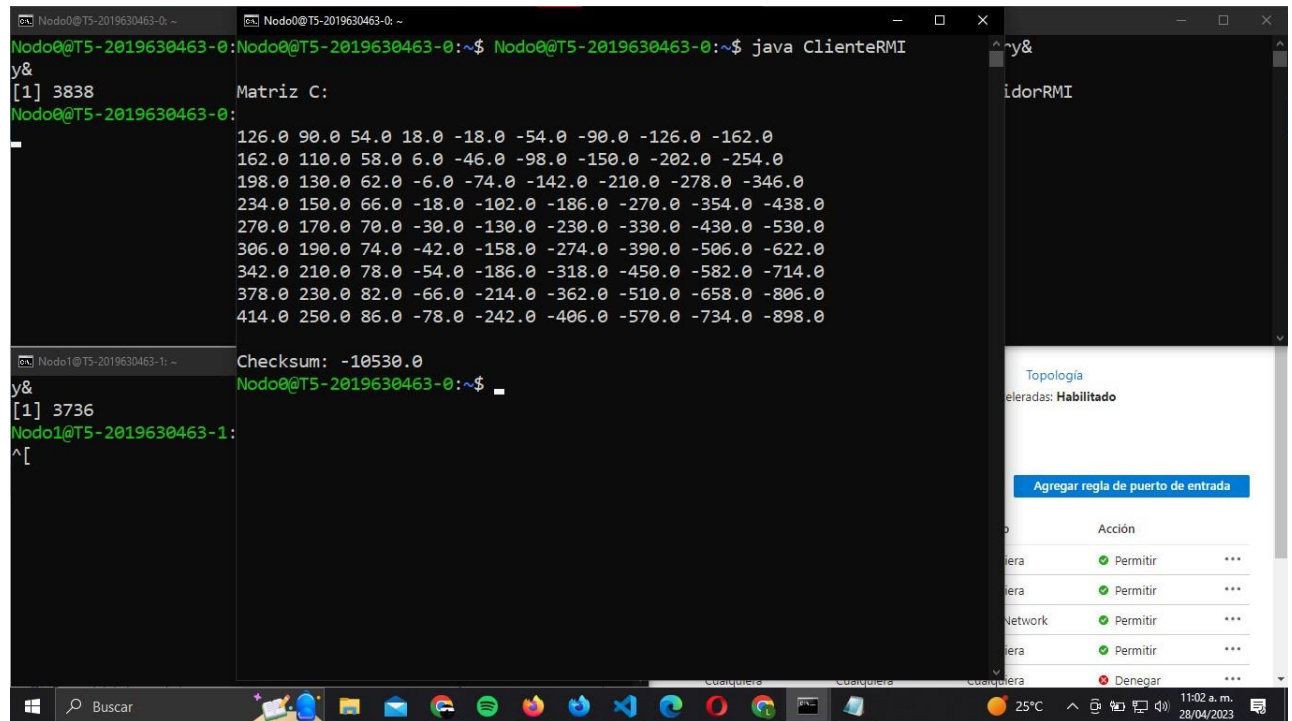


Figura 2.5. Ejecución de rmiregistry& y ServidorRMI.java en máquinas virtuales.

Ahora, para la ejecución del ClienteRMI.java en el nodo 0, se debe abrir otra terminal del Nodo 0, una vez abierta, como ya se ha compilado el programa ClienteRMI.java, solo hay que ejecutarlo.

Primero se ejecuta con el caso donde $N=9$ y $M=4$.

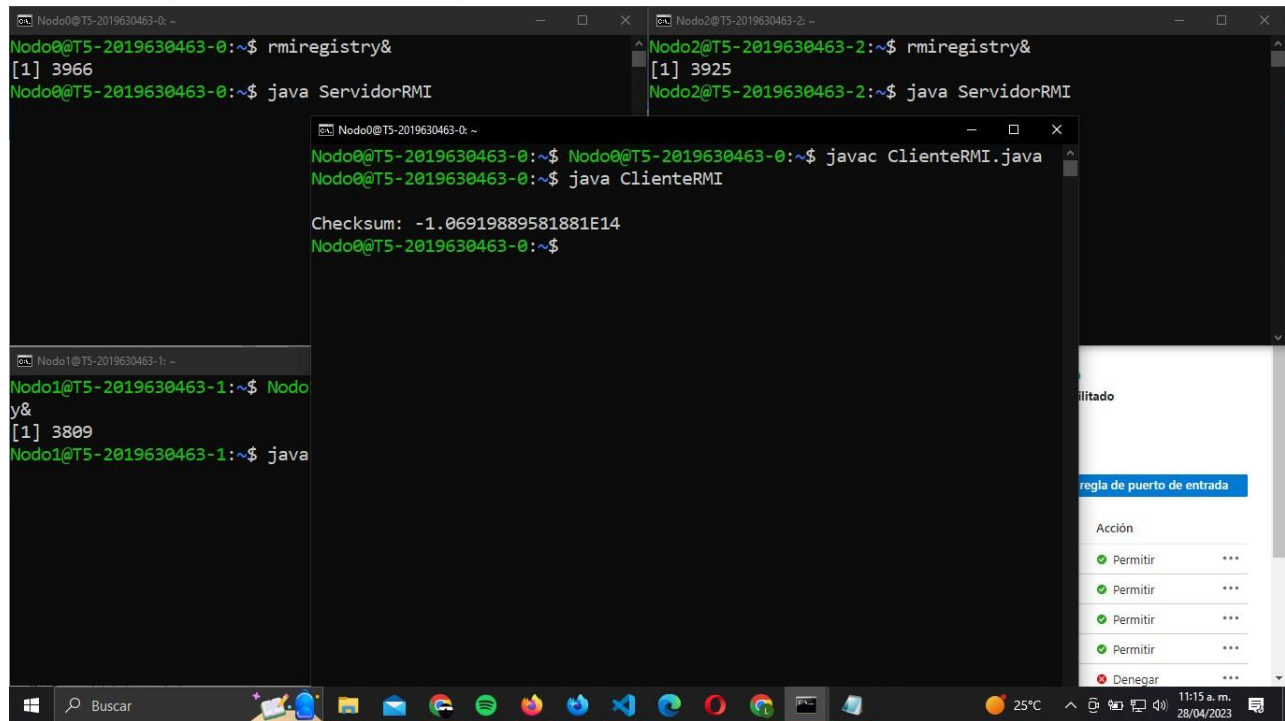
Obteniendo la siguiente Matriz C y el checksum correspondiente.



```
Nodo0@T5-2019630463-0: ~  
Nodo0@T5-2019630463-0:~$ java ClienteRMI  
y&  
[1] 3838  
Nodo0@T5-2019630463-0:~$  
Matriz C:  
126.0 90.0 54.0 18.0 -18.0 -54.0 -90.0 -126.0 -162.0  
162.0 110.0 58.0 6.0 -46.0 -98.0 -150.0 -202.0 -254.0  
198.0 130.0 62.0 -6.0 -74.0 -142.0 -210.0 -278.0 -346.0  
234.0 150.0 66.0 -18.0 -102.0 -186.0 -270.0 -354.0 -438.0  
270.0 170.0 70.0 -30.0 -130.0 -230.0 -330.0 -430.0 -530.0  
306.0 190.0 74.0 -42.0 -158.0 -274.0 -390.0 -506.0 -622.0  
342.0 210.0 78.0 -54.0 -186.0 -318.0 -450.0 -582.0 -714.0  
378.0 230.0 82.0 -66.0 -214.0 -362.0 -510.0 -658.0 -806.0  
414.0 250.0 86.0 -78.0 -242.0 -406.0 -570.0 -734.0 -898.0  
Checksum: -10530.0  
Nodo0@T5-2019630463-0:~$  
Nodo1@T5-2019630463-1: ~  
y&  
[1] 3736  
Nodo1@T5-2019630463-1:~$  
^
```

Figura 2.6. Cálculo de matriz C y checksum en el nodo 0 para el caso $N=9$ y $M=4$.

Para el caso $N=900$ y $M=400$, no se despliega la matriz C mostrando solo el checksum.



```
Nodo0@T5-2019630463-0: ~$ rmiregistry&
[1] 3966
Nodo0@T5-2019630463-0: ~$ java ServidorRMI

Nodo2@T5-2019630463-2: ~$ rmiregistry&
[1] 3925
Nodo2@T5-2019630463-2: ~$ java ServidorRMI

Nodo0@T5-2019630463-0: ~$ javac ClienteRMI.java
Nodo0@T5-2019630463-0: ~$ java ClienteRMI

Checksum: -1.06919889581881E14
Nodo0@T5-2019630463-0: ~$

Nodo1@T5-2019630463-1: ~$ rmiregistry&
[1] 3809
Nodo1@T5-2019630463-1: ~$ java
```

regla de puerto de entrada

Acción
Permitir ***
Permitir ***
Permitir ***
Permitir ***
Denegar ***

25°C 11:15 a. m. 28/04/2023

Figura 2.7. Cálculo de matriz C y checksum en el nodo 0 para el caso $N=900$ y $M=400$.

Conversación con Chat GPT.

Luis Antonio Flores Castro.

Buen día ¿me podrías ayudar a realizar lo siguiente por favor?

Hacer en Java RMI lo siguiente.

Se crearán 4 archivos de tipo java RMI.

Solo se debe crear el archivo "ClienteRMI" por el momento.

Dentro del archivo "ClienteRMI" se deberá realizar lo siguiente:

Inicializar una matriz A de dimensiones [N][M] de la siguiente forma:

$A[i][j] = 2*i + 3*j$

Inicializar una matriz B de dimensiones [M][N] de la siguiente forma:

$B[i][j] = 3*i - 2*j$

Se deberá crear una matriz llamada BT la cual será igual a la matriz transpuesta de la matriz B.

Las matrices A,B y BT deben ser de tipo float.

Ahora se deberá dividir la matriz A y BT en nueve partes iguales, es decir, si la matriz A tiene dimensiones [9][4] por ejemplo, deberán generarse 9 submatrices de tamaño [1][4], y para la matriz BT realizar el mismo procedimiento, si tiene dimensiones de [9][4] deberán generarse 9 submatrices de tamaño [1][4]

Se declarará ahora una matriz Caux de tipo float, esta será una matriz tridimensional donde se guardarán 81 submatrices del tamaño [N/9][N/9].

Chat GPT.

¡Hola! Claro que puedo ayudarte a crear el archivo "ClienteRMI" en Java RMI para realizar las operaciones que necesitas. Aquí te dejo el código:

```
import java.rmi.*;
```

```
import java.rmi.server.*;
```

```
import java.util.Arrays;
```

```
public class ClienteRMI extends UnicastRemoteObject implements ClienteInterfaz {
```

```
    private int N = 9;
```

```
    private int subN = N/9;
```

```

private int M = 4;
private float[][] A = new float[N][M];
private float[][] B = new float[M][N];
private float[][] BT = new float[N][M];
private float[][][] Caux = new float[81][N/9][N/9];

public ClienteRMI() throws RemoteException {
    super();
    // Inicializar matriz A
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            A[i][j] = 2*i + 3*j;
        }
    }
    // Inicializar matriz B
    for (int i = 0; i < M; i++) {
        for (int j = 0; j < N; j++) {
            B[i][j] = 3*i - 2*j;
        }
    }
    // Obtener matriz BT
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            BT[i][j] = B[j][i];
        }
    }
    // Dividir matrices A y BT en submatrices
    int k = 0;
    for (int i = 0; i < N; i += N/9) {
        for (int j = 0; j < M; j += M/3) {
            float[][] subA = Arrays.copyOfRange(A, i, i+N/9);

```

```

        float[][] subBT = Arrays.copyOfRange(BT, i, i+N/9);
        for (int x = 0; x < subA.length; x++) {
            subA[x] = Arrays.copyOfRange(subA[x], j, j+M/3);
            subBT[x] = Arrays.copyOfRange(subBT[x], j, j+M/3);
        }
        Caux[k++] = subA;
        Caux[k++] = subBT;
    }
}

```

```

public static void main(String[] args) {
    try {
        ClienteInterfaz cliente = new ClienteRMI();
        System.out.println("Matriz A:");
        imprimirMatriz(cliente.getMatrizA());
        System.out.println("Matriz BT:");
        imprimirMatriz(cliente.getMatrizBT());
        System.out.println("Submatrices de A y BT:");
        for (int i = 0; i < 81; i++) {
            System.out.println("Submatriz " + (i+1) + ":");
            imprimirMatriz(cliente.getSubmatriz(i));
        }
    } catch (Exception e) {
        System.err.println("Error en el cliente: " + e.toString());
    }
}

```

```

public static void imprimirMatriz(float[][] matriz) {
    for (int i = 0; i < matriz.length; i++) {
        for (int j = 0; j < matriz[0].length; j++) {

```

```

        System.out.print(matriz[i][j] + " ");
    }
    System.out.println();
}
System.out.println();
}

```

Luis Antonio Flores Castro.

Muy bien, ahora se creará el segundo archivo llamado "ClaseRMI", dentro de este archivo deberá implementarse un método "multiplicarMatrices()", el cual recibirá 4 parametros, los dos primeros serán dos matrices de tipo float, y los otros dos, una variable N y otra variable M ambas de tipo int. El método realizará lo siguiente:

multiplicar dos matrices de punto flotante de tamaño NxM y MxN (representadas como los arreglos de dos dimensiones A y B, respectivamente) para producir una nueva matriz de tamaño NxN (representada como el arreglo de dos dimensiones aux).

La operación se debe realizar mediante tres ciclos anidados que recorren todas las filas y columnas de la matriz auxiliar. En cada iteración, se calcula el producto punto a punto de cada fila de A con cada columna de B, lo que implica un total de M operaciones por iteración.

El resultado de estas operaciones se almacena en el arreglo auxiliar en la posición correspondiente, y finalmente se devuelve la matriz auxiliar completa.

Chat GPT.

¡Claro! Aquí te dejo el archivo en Java que realiza la multiplicación de matrices que has descrito dentro del archivo ClaseRMI:

```

import java.rmi.*;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class ClaseRMI extends UnicastRemoteObject implements InterfaceRMI {
    private static final long serialVersionUID = 1L;

    public ClaseRMI() throws RemoteException {
        super();
    }
}

```



```

    public float[][] multiplicarMatrices(float[][] A, float[][] B, int N, int M)
    {
        float[][] aux = new float[N][N];
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                int op = 0;
                for (int k = 0; k < M; k++) {
                    op += A[i][k] * B[j][k];
                }
                aux[i][j] = op;
            }
        }
        return aux;
    }
}

```

Luis Antonio Flores Castro.

Muy bien, ahora se creará el archivo "InterfaceRMI" en el cual se deberá mandar llamar el método "multiplicarMatrices()" que ya ha sido implementado anteriormente.

ChatGPT.

¡Claro que sí! Aquí te dejo el código de la interfaz de Java InterfaceRMI:

```

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface InterfaceRMI extends Remote {
    public float[][] multiplicarMatrices(float[][] A, float[][] B, int N, int M)
    throws RemoteException;
}

```

Luis Antonio Flores Castro.

Ahora, ¿Podrías realizar lo siguiente por favor? Se deberá crear un archivo llamado "ServidorRMI"

La clase ServidorRMI deberá tener un método main que toma un argumento de cadena de texto. Dentro del método main, se establece la url del objeto remoto en el formato "rmi://localhost/ServidorRMI", donde "localhost" es el nombre del host y "ServidorRMI" es el

nombre que se le asigna al objeto remoto. Se utilizará el método estático `rebind()` de la clase `Naming` para registrar la instancia de `ClaseRMI` en el `rmiregistry`.

ChatGPT.

¡Por supuesto! El programa `ServidorRMI` se podría implementar de la siguiente forma:

```
import java.rmi.Naming;

public class ServidorRMI {

    public static void main(String[] args) throws Exception {

        String url = "rmi://localhost/ServidorRMI";

        ClaseRMI obj = new ClaseRMI();

        // Uso de Naming

        Naming.rebind(url, obj);

    }

}
```

Luis Antonio Flores Castro.

Muy bien, necesito que dentro este programa `ClienteRMI` ahora se debe crear una clase llamada `Hilos` que extiende de `Thread`, dentro de ella se deberán declarar 4 variables, la primera variable deberá ser `InterfaceRMI` servidor, la segunda la variable identificador de tipo `int`, la tercera la variable `int N` y la ultima la variable `int M`. Se deberá implementar un constructor con las variables que se mencionaron anteriormente. después en el método `run()` de esta clase ahora deberá implementarse 3 if anidados, si el identificador es igual a 0 se deberá llamar al método "`multiplicarMatrices()`" el cual ya ha sido realizado en un archivo `ClaseRMI` y declarado en otro archivo llamado `InterfaceRMI` por lo cual no se tiene que hacer, la matriz `Caux`, es una matriz tridimensional estatica de `[81][N/9][N/9]` por lo cual dentro de ciclos anidados se deberá mandar llamar al método "`multiplicarMatrices()`", el cual tiene 4 parametros, el primero una matriz `Float`, el segundo otra matriz `Float`, el tercero una variable `N`, y el otro una variable `M` ambas de tipo `int`, dentro de la matriz `Caux`, se deberán guardar 27 matrices, cada matriz devuelta por el metodo "`multiplicarMatrices()`" los parametros de este metodo iran iterando, primero se debera pasar la matriz `subA[0]` e ir iterando con la matriz `subB[0]`, empezando desde 0 hasta llegar al indice 9 de la matriz `subB`, cuando se llegue a este indice ahora el indice de `subA` cambia a `[1]` y se repite el proceso, esto se repetira hasta llegar al indice `[2]` de la matriz `subA`.

Chat GPT.

Entiendo que necesitas implementar una clase llamada `Hilos` dentro del programa `ClienteRMI`, la cual debe extender de `Thread` y declarar las siguientes variables:

servidor: de tipo `InterfaceRMI`

identificador: de tipo `int`

N: de tipo int

M: de tipo int

Para esto, puedes utilizar el siguiente código como base:

```
public class Hilos extends Thread {  
    private InterfaceRMI servidor;  
    private int identificador;  
    private int N;  
    private int M;  
  
    public Hilos(InterfaceRMI servidor, int identificador, int N, int M) {  
        this.servidor = servidor;  
        this.identificador = identificador;  
        this.N = N;  
        this.M = M;  
    }  
  
    public void run() {  
        if (identificador == 0) {  
            // Llamar al método "multiplicarMatrices()"   
        } else if (identificador == 1) {  
            // Implementar lógica para identificador 1  
        } else if (identificador == 2) {  
            // Implementar lógica para identificador 2  
        }  
    }  
}
```

Dentro del método run(), se deben implementar los if necesarios para realizar la tarea correspondiente según el valor de la variable identificador.

Si identificador es igual a 0, se debe llamar al método "multiplicarMatrices()" que ya ha sido implementado en la clase ClaseRMI y declarado en la interfaz InterfaceRMI. Para llamar a este método, puedes utilizar el siguiente código:

```

if (identificador == 0) {
    try {
        float[][] matrizA = new float[N][N];
        float[][] matrizB = new float[N][N];

        // Inicializar las matrices matrizA y matrizB
        float[][] matrizC = servidor.multiplicarMatrices(matrizA, matrizB, N, M);

        // Procesar la matrizC resultante
    } catch (RemoteException e) {
        e.printStackTrace();
    }
}

```

La matriz Caux mencionada en la descripción es una matriz tridimensional estática de [81][N/9][N/9], por lo que se puede inicializar de la siguiente forma:

```
float[][][] Caux = new float[81][N/9][N/9];
```

Dentro de ciclos anidados, se deben llamar al método "multiplicarMatrices()" 27 veces y guardar el resultado en la matriz Caux. Para esto, puedes utilizar el siguiente código como base:

```

if (identificador == 1) {
    int index = 0;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            for (int k = 0; k < 3; k++) {
                float[][] subA = new float[N/3][N/3];
                float[][] subB = new float[N/3][N/3];

                // Inicializar las matrices subA y subB
                float[][] subC = servidor.multiplicarMatrices(subA, subB, N/3,
M);

                // Guardar el resultado en la matriz Caux
                Caux[index++] = subC;
            }
        }
    }
}

```

Este código itera sobre las 27 submatrices de tamaño $N/3$ de las matrices A y B, llama al método "multiplicarMatrices()" y guarda el resultado en la matriz Caux.

Luis Antonio Flores Castro.

Necesito llenar la siguiente matriz `static float[][][] Caux = new float[81][subN][subN];` llamando al método `multiplicarMatrices`, esto puede ser de la siguiente manera `Caux[][][] = multiplicarMatrices(subA[], subB[], N, M)`, la matriz Caux almacena 81 matrices de $[N/9][N/9]$, cada que se llama al método `multiplicarMatrices`, este devuelve una matriz de tamaño $[N/9][N/9]$ por lo que se irían guardando en Caux, ahora necesito que se llama el método 27 veces, dentro de esas 27 veces se debe llamar al método de la siguiente manera `multiplicarMatrices(subA[indice1], subB[indice2], N, M)` donde `indice1` es igual a 0 en un comienzo e `indice2` va iterando desde 0 hasta 9, cuando se llegue a 9 ahora `indice1` cambia a 1 e `indice2` vuelve a iterar desde 0 hasta 9, cuando llega nuevamente a 9, ahora `indice1` cambia a 2 e `indice2` vuelve a iterar desde 0 hasta 9, de esta forma obteniendo las 27 llamadas al método, podrías implementarlo por favor?

Chat GPT.

Sí, por supuesto. A continuación te muestro cómo quedaría el método `run()` de la clase `Hilos` que extiende de `Thread` para realizar las operaciones descritas:

```
public class Hilos extends Thread {
    private InterfaceRMI servidor;
    private int identificador;
    private int N;
    private int M;

    public Hilos(InterfaceRMI servidor, int identificador, int N, int M) {
        this.servidor = servidor;
        this.identificador = identificador;
        this.N = N;
        this.M = M;
    }

    public void run() {
        if (identificador == 0) {
            static float[][][] Caux = new float[81][N/9][N/9];
```

```

        for (int i = 0; i < 9; i++) {
            for (int j = 0; j < 9; j++) {
                Caux[i][j] = servidor.multiplicarMatrices(subA[i], subB[j],
N, M);
            }
        }
    } else if (identificador > 0 && identificador < 28) {
        int indice1 = (identificador - 1) / 9;
        int indice2 = (identificador - 1) % 9;

        static float[][] matrizResultado =
servidor.multiplicarMatrices(subA[indice1], subB[indice2], N, M);

        // Aquí puedes hacer lo que necesites con la matrizResultado obtenida
    } else {
        System.out.println("Identificador no válido");
    }
}
}
}

```

En el primer if se crea la matriz Caux de $[81][N/9][N/9]$ y se itera sobre subA y subB para llamar al método multiplicarMatrices y guardar el resultado en la matriz Caux.

En el segundo if se obtienen los índices de subA y subB que corresponden al identificador del hilo actual, se llama al método multiplicarMatrices y se guarda el resultado en la matriz matrizResultado (que en este caso es de tamaño $[N/9][N/9]$), y finalmente se puede hacer lo que necesites con la matriz resultado.

Recuerda que antes de ejecutar los hilos debes haber inicializado las matrices subA y subB con los valores correspondientes. Además, la implementación anterior asume que el identificador de cada hilo será un número entero entre 0 y 27, inclusive. Si necesitas un rango de identificadores diferente, simplemente modifica las condiciones en los if según tus necesidades.

Luis Antonio Flores Castro.

La instrucción anterior que te di debe de ir todo dentro del if identificador==0, y quiero que repitas esta instrucción cuando el identificador es igual a 1, solo que el ciclo for se iteraría desde 27 hasta 54, realizar la misma instrucción para cuando el identificador es igual a 2, con el ciclo for iterando desde 54 hasta 81.

Chat GPT.

Perfecto, aquí te dejo el código actualizado del método run() de la clase Hilos que cumple con las especificaciones que me proporcionaste:


```

try {

    int indice1 = 0;
    int indice2 = 0;
    if (identificador == 0) {
        // Iterar sobre la primera dimensión de Caux
        for (int i = 0; i < 27; i++) {
            // Llamar al método multiplicarMatrices para obtener una
matriz de [N/9][N/9]
            float[][] subC =
servidor.multiplicarMatrices(subA[indice1], subBT[indice2], N, M);

            // Iterar sobre la matriz subC y copiar los valores en
la matriz Caux
            for (int j = 0; j < subC.length; j++) {
                for (int k = 0; k < subC[j].length; k++) {
                    Caux[i][j][k] = subC[j][k];
                }
            }

            // Actualizar los índices de subA y subB
            indice2++;
            if (indice2 == 9) {
                indice2 = 0;
                indice1++;
            }
        }

    } else if (identificador == 1) {
        indice1 = 3;
        indice2 = 0;
        // Implementar lógica para identificador 1
        for (int i = 27; i < 54; i++) {

```

```

// Llamar al método multiplicarMatrices para obtener una
matriz de [N/9][N/9]

float[][] subC =
servidor.multiplicarMatrices(subA[indice1], subBT[indice2], N, M);

// Iterar sobre la matriz subC y copiar los valores en
la matriz Caux

for (int j = 0; j < subC.length; j++) {
    for (int k = 0; k < subC[j].length; k++) {
        Caux[i][j][k] = subC[j][k];
    }
}

// Actualizar los índices de subA y subB
indice2++;
if (indice2 == 9) {
    indice2 = 0;
    indice1++;
}
}

} else if (identificador == 2) {
    // Implementar lógica para identificador 2
    indice1 = 6;
    indice2 = 0;
    // Implementar lógica para identificador 1
    for (int i = 54; i < 81; i++) {
        // Llamar al método multiplicarMatrices para obtener una
matriz de [N/9][N/9]

float[][] subC =
servidor.multiplicarMatrices(subA[indice1], subBT[indice2], N, M);

// Iterar sobre la matriz subC y copiar los valores en
la matriz Caux

```

```

        for (int j = 0; j < subC.length; j++) {
            for (int k = 0; k < subC[j].length; k++) {
                Caux[i][j][k] = subC[j][k];
            }
        }

        // Actualizar los índices de subA y subB
        indice2++;
        if (indice2 == 9) {
            indice2 = 0;
            indice1++;
        }
    }
}

} catch (Exception e) {
    System.out.println(e.getMessage());
}

```

En este código, primero se verifica si `identificador` es igual a 0. Si es así, se procede a crear una matriz tridimensional `Caux` de tamaño `[81][N/subN][N/subN]`. Luego, se inicia un ciclo `for` que iterará 27 veces (ya que necesitamos 27 llamadas al método `multiplicarMatrices()`).

Dentro del ciclo `for`, se llama al método `multiplicarMatrices()` pasándole como parámetros los valores correspondientes de `subA` e `subB`, así como `N` y `M`. El resultado se almacena en la posición correspondiente de la matriz `Caux`.

También hay dos variables `indice1` e `indice2`, que se utilizan para obtener las matrices correspondientes de `subA` y `subB` en cada iteración del ciclo `for`. `indice1` comienza en 0 y aumenta cada vez que `indice2` alcanza el valor 9. `indice2` comienza en 0 y se incrementa en cada iteración del ciclo `for`.

Conclusiones.

Mediante la realización de esta tarea pude terminar de entender la funcionalidad de Java RMI. Al empezar a desarrollar esta tarea me generaba ciertas dudas acerca de cómo se compartía el objeto ya que había visto ejemplos de como funcionaban los programas de manera local, pero no mediante una conexión con otros equipos con direcciones diferentes. De los retos que tuve fue el desarrollar el archivo Cliente RMI con Chat GPT debido a que agregaba ciertos métodos que no le había indicado y afectaban un poco el funcionamiento. Otro punto que busque optimizar fue la declaración y acceso a las matrices, al inicio Chat GPT me generó una versión de código donde las matrices y ciertas variables eran declaradas dentro de los métodos, pero al indicarle que estas se pasarían como parámetros a distintos métodos lo cambio a una declaración global, para el mejor acceso, noté que mediante esta implementación fue mucho mejor por que la cantidad de líneas se redujo en muy buena forma. Por otro lado, en cuanto a las operaciones no hubo mucho problema puesto que ya se habían realizado estas operaciones en anteriores tareas y los prompts eran muy similares por lo que al redactarlos no hubo mayor problema, en el método para multiplicar matrices de igual forma como ya se había realizado ya se contaba con la experiencia de como indicarle al Chat GPT que no era un método que implementaba la multiplicación de matrices si no se hacia de otra forma. Donde si hubo un poco de análisis y tomo mas tiempo fue el acceder a la matriz final C, el Chat GPT desde un inicio la declaró como un cubo por lo cual al momento de acomodar las demás matrices dentro de ella en ocasiones se tomaban los índices de forma incorrecta, generando algunas inconsistencias en los datos. Una vez terminados los programas, al momento de compilarlos y ejecutarlos en las maquinas virtuales de azure tuve un problema, dentro del archivo Cliente RMI había colocado las IP's públicas de dichas maquinas por lo cual no se generaba ningún resultado, solo se quedaba esperando y se terminaba la ejecución de dichos programas debido a que el tiempo de conexión se había agotado, sin embargo, recordé que se tenia que realizar la conexión con las IP's privadas de cada máquina, así que se configuró y funcionó de manera correcta.

Por último, puedo decir que me fue un poco complicada el desarrollar la tarea, ya que como dije al inicio me generaba ciertas dudas la conexión por ese motivo me tomo un poco más de tiempo redactar de forma más clara los prompt para Chat GPT, ya una vez entendido bien el funcionamiento no tuve mayores problemas en la aplicación y ejecución.